Structural Information Aided Automated Test Method for Magic 4GL

Ferenc Horváth, Richárd Dévai, Tamás Gergely

Nowadays testing data intensive, GUI enhanced applications (like those designed with Magic 4GL) properly on an easily maintainable way has become a more crucial part of the application life cycle. For 3GL applications there are many evolving technologies to support automatized GUI testing for web applications and also for their desktop counterparts.

The first generation GUI tester tools were layout dependent and more like simple recorder and player tools which could replay the users' earlier actions. The main drawbacks of this solution were the poor verifiability and the sensitiveness to even the smallest modifications of the layout. Also, these tools needed to be controlled by user interactions.

In the next generation the tools got more sophisticated by supporting the identification of GUI elements in layout independent ways. This identification allowed the record or generation of test cases, as the interactions on the GUI elements took place based on a general model instead of layout positions. Based on previous works [2] and [3] we aimed to create a new test generator system. We created a new path and test script generator tool which can provide a sophisticated and simple way to create interpretable test cases for the used test execution tool.

Path generation is a very complex task because one has to face obstacles like the exponential relation between the number of branches and the number of the possible paths, or the presence of loops in every real program which induces the number of paths to converge to infinity [1]. Therefore robust algorithms are needed.

Our path generation strategy is based on well known algorithms like breadth-first and depth-first search. Considering the time and space complexities, these algorithms perform well enough to allow us to build upon them while developing our advanced strategies. However, we have to introduce some modifications to meet the requirements of our domain. These modifications affect the generation process in numerous ways. For example, we created an interface that can be used to build a business model for the program being tested. This model defines constrains which will be used by the generator algorithm as a form of selection method to narrow down the set of used program subcomponents (e.g. tasks which represent main functional components in Magic 4GL). This way the set of the generated paths can be reduced significantly and contains only the most relevant ones.

Path generation strategy is a crucial part of the test script generation. After the selection of the appropriate path set, we have to convert the information we gained into interpretable test scripts. The number of the generated paths has a strong influence on the final number of test scripts as well as on the strategy we use to determine how we select and permute values among different variables.

The co-domain for each variable is based on its type and control flow context. After we extract the influencing data properties (D/U information), we have to narrow down the co-domain of a variable based on the influencing expressions and statements which appear on the corresponding path. Value selection on co-domains is also a crucial part of the generation as it can cause an enormous growth of the number of test scripts. Choosing a good strategy for solving the problems we listed above is essential in order to get an accurate and usable test script set as the output of the described script generator algorithm. Only a small enough test set size can be manageable even if the pre-configuration of the Magic xpa program and the execution of the scripts are totally automated.

We present a schematic model of the system we built to test Magic xpa applications in Figure 3. This system is based on the work of Dévai et al. [2] to gain usable behavioral information about the program under test and Fritsi et al. [3] to run our assembled scripts. We connected

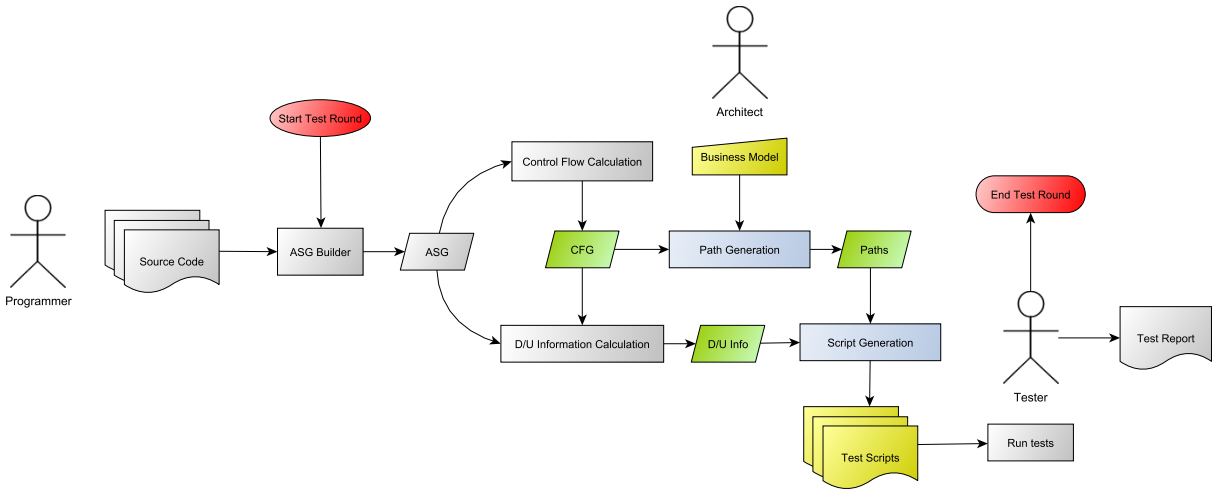

Figure 3: Flowchart of Magic test process

this two components by developing a path and a script generator that process behavioral information and assemble executable scripts for the test runner.

In conclusion, we would like to present our path and script generator algorithms and their implementations for Magic xpa applications. In addition, we intend to demonstrate how these components cooperate with the existent solutions and as a possible use of the results we introduce a test method that has been completed by the application of our path and script generator tools.

Acknowledgements

This research was supported by the Hungarian national grant GOP-1.1.1-11-2011-0039.

References

[1] T. Bakota, Á Beszedes, T. Gergely, M. Gyalai, T. Gyimóthy, and D. Füleki. Semi-automatic test case generation from business process models. In *Proceedings of the 11th Symposium on Programming Languages and Software Tools (SPLST'09) and 7th Nordic Workshop on Model Driven Software Engineering (NW-MODE'09)*, pages 5–18, Tampere, Finland, August 26-28 2009.

[2] Richárd Dévai, Judit Jász, Csaba Nagy, and Rudolf Ferenc. Designing and implementing control flow graph for Magic 4th generation language. In Ákos Kiss, editor, *Proceedings of the 13th Symposium on Programming Languages and Software Tools (SPLST'13)*, pages 200–214, Szeged, Hungary, 2013. University of Szeged.

[3] Dániel Fritsi, Csaba Nagy, Rudolf Ferenc, and Tibor Gyimóthy. A layout independent GUI test automation tool for applications developed in Magic/uniPaaS. In *Proceedings of the 12th Symposium on Programming Languages and Software Tools (SPLST'11)*, pages 248–259, 2011.